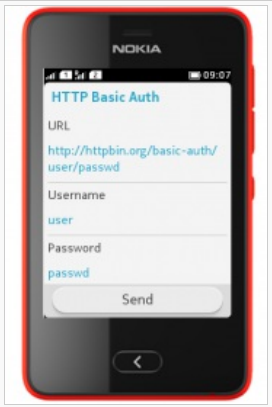
HTTP basic access authentication in Java ME

This article explains how to access HTTP resources protected by a basic access authentication in Java ME.

Introduction

Basic access authentication is the simplest form of authorization available for HTTP resources.



Java app accessing resources via HTTP basic access authentication

The typical process involved when accessing a HTTP resource protected by this authentication method is the following:

1. the client sends an unauthenticated request to the HTTP server
2. the HTTP server replies with a *401 status code*, meaning that the client is unauthorized to access the resource, and adding an HTTP header named `WWW-Authenticate`. This header specifies the authentication method and the realm the user should authenticate for.
3. the client reads the `WWW-Authenticate` header, and sends a new authenticated request by adding the `Authorization` HTTP header. This header contains the user credentials formatted as *<USERNAME>:<PASSWORD>* and Base64 encoded.

Note: If the authentication method and realm used by the HTTP server is already known, the client can also skip the first two steps, and send immediately an authenticated request as specified by step 3.

Implementation

The Java app sends a first request to the HTTP server, without any authentication credential.

If the server replies with a 401 HTTP status code, identified in Java by the `HttpConnection.HTTP_UNAUTHORIZED` constant, the Java app checks the *WWW-Authenticate* HTTP header, verifying if it starts with the *Basic* text: that string identifies the basic access authentication method.

```
HttpConnection hc = (HttpConnection) Connector.open(url);

int responseCode = hc.getResponseCode();

if(responseCode == HttpConnection.HTTP_UNAUTHORIZED)
{
 String wwwAuthHeader = (String)hc.getHeaderField("WWW-Authenticate");

 if(wwwAuthHeader != null && wwwAuthHeader.indexOf("Basic ") == 0)
 {
  // HTTP basic access authentication must be used to access this resource
 }
}
```

When the Java app identifies the basic access authentication method, it can also check the realm it should authenticate for, by accessing the remaining portion of the *WWW-Authenticate* header.

Once the Java app is ready, it must send a second HTTP request specifying the *Authorization* header, containing the base64 encoded version of the user credentials, formatted as *<USERNAME>:<PASSWORD>*. Since Java ME has no inbuilt implementation of the Base64 encoding algorithm, this must be implemented with custom code. A possible implementation of the algorithm is available here, and is reported below for ease of use:

```
    String base64Encode(String s) {

        // the result/encoded string, the padding string, and the pad count
        String r = "", p = "";
        int c = s.length() % 3;

        // add a right zero pad to make this string a multiple of 3 characters
        if (c > 0) {
            for (; c < 3; c++) {
                p += "=";
                s += "\0";
            }
        }
```


(C) Copyright Nokia 2013. All rights reserved.

```
        // increment over the length of the string, three characters at a time
        for (c = 0; c < s.length(); c += 3) {

            // we add newlines after every 76 output characters, according to
            // the MIME specs
            if (c > 0 && (c / 3 * 4) % 76 == 0)
                r += "\r\n";

            // these three 8-bit (ASCII) characters become one 24-bit number
            int n = (s.charAt(c) << 16) + (s.charAt(c + 1) << 8)
                    + (s.charAt(c + 2));

            // this 24-bit number gets separated into four 6-bit numbers
            int n1 = (n >> 18) & 63, n2 = (n >> 12) & 63, n3 = (n >> 6) & 63, n4 = n & 63;

            // those four 6-bit numbers are used as indices into the base64
            // character list
            r += "" + base64chars.charAt(n1) + base64chars.charAt(n2)
                    + base64chars.charAt(n3) + base64chars.charAt(n4);
        }

        return r.substring(0, r.length() - p.length()) + p;
    }
```

By using the above method, the Java app can properly encode the user credentials, and send a second HTTP request as follows:

```
String authorizationHeader= "Basic " + base64Encode(username + ":" + password);

HttpConnection hc = (HttpConnection) Connector.open(url);

hc.setRequestProperty("Authorization", authorizationHeader);

responseCode = hc.getResponseCode();
```

In this case, if the provided user credentials are correct, the HTTP server will reply with a *200 status code*.

Correct authenticated request receives a 200 HTTP response code

Testing

Multiple test servers are available for checking the implementation of the HTTP basic access authentication. One of those is available at the following address: http://httpbin.org/basic-auth/{USERNAME}/{PASSWORD} , where {USERNAME} and {PASSWORD} can be customized with the desired values, and must be used as credentials for the *Authorization* header.

The sample Java app attached to this article uses that testing server, but data can be easily changed in order to test against other testing or production servers.

Summary

This article illustrates a possible Java ME implementation of the basic access authentication method used for HTTP requests.

Full source code of the sample Java app illustrated in this article is available here: Media:WikiHttpBasicAuth.zip

(C) Copyright Nokia 2013. All rights reserved.

(C) Copyright Nokia 2013. All rights reserved.